

Software Requirements Specification

| | |
|------------------|--------------|
| File Name | Photo Studio |
|------------------|--------------|

| | |
|----------------------|--------------------|
| Creation Date | September 19. 2025 |
|----------------------|--------------------|

| | |
|--------------------|-------------------------------------------|
| Authored by | Ringo-Lehari Purge, Martí Pujol I Saumell |
|--------------------|-------------------------------------------|

| | |
|--------------------|--|
| Reviewed by | |
|--------------------|--|

REVISION HISTORY

| VERSION | DATE | DESCRIPTION | AUTHOR |
|---------|------|-------------|--------|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

CONTENTS

| | | |
|----------|--------------------------------------------------------|----------|
| 1 | SW SYSTEM OVERVIEW..... | 4 |
| 1.1 | PURPOSE..... | 4 |
| 1.2 | SCOPE..... | 4 |
| 1.3 | USE-CASE DIAGRAM..... | 4 |
| 1.4 | GENERAL CONSTRAINTS..... | 4 |
| 1.5 | ASSUMPTIONS AND DEPENDENCIES..... | 4 |
| 1.6 | ACRONYMS AND ABBREVIATIONS..... | 4 |
| 2 | SW FUNCTIONAL REQUIREMENTS..... | 5 |
| 2.1 | FEATURES / FUNCTIONS TO BE IMPLEMENTED..... | 5 |
| 2.1 | ACCEPTANCE CRITERIA..... | 5 |
| 2.2 | IMPLEMENTATION REQUIREMENTS..... | 5 |
| 3 | SW NON-FUNCTIONAL REQUIREMENTS..... | 6 |
| 3.1 | RESOURCE CONSUMPTION..... | 6 |
| 3.2 | LICENSE ISSUES..... | 6 |
| 3.3 | CODING STANDARD..... | 6 |
| 3.4 | MODULAR DESIGN..... | 6 |
| 3.5 | RELIABILITY..... | 6 |
| 3.6 | PORTABILITY..... | 6 |
| 3.7 | GENERAL OPERATIONAL GUIDELINES..... | 6 |
| 4 | SW DESIGN ARTIFACTS..... | 7 |
| 4.1 | CRC CARDS (CLASS-RESPONSIBILITY-COLLABORATION)..... | 7 |
| 4.2 | CONCEPTUAL UML DIAGRAM (ENTITIES & RELATIONSHIPS)..... | 7 |

1 SW System Overview

Specify the purpose and the overview of the SRS.

This SRS describes requirements of a software system to be used in Photo Studios where photos are printed and film developed.

The primary goals are to speed up the distribution of the orders, optimize employees' workflow and to give the studio administrator an overview of the work done, the income and the expenses (material used).

1.1 Purpose

Describe the purpose of the system. What problem does it solve? Who are the intended users? Why is it being developed?

The main purpose of this software is to optimize the workflow of the Photo Studio project.

The software system will also record orders in a centralized manner, track their completion and the related income and expenses for logging and to give a revenue report to the studio administrator.

The system is meant to allow the receptionist to:

- Create clients in the system
- Easily create orders with the client data.
- Generate automatically two types of orders (Client and photographer)
- Easy access to client orders.
- Automatically calculate the price (meaning the surcharge) and mark an express order automatically.
- Have centralized mappings with all the orders recorded.
- Create and submit the daily revenue reports easily.

The system will allow the photographer to:

- Submit a report on consumed materials at the end of the day.
- Submit the completion of an order.

The intended users are:

- the client
- the receptionist
- the photographer
- the studio administrator

1.2 Scope

Define the scope of the system. What functionality is included? What is explicitly excluded? Mention benefits and key features.

The system includes:

- Creation of clients.
- Creation of the orders.
- The transmission of orders (receptionist -> Photographer, receptionist -> Client).
- Price calculation based on order deadline (Mark it as express if necessary).
- Track of the consumables materials.

- Submissions of orders completed by the photographer.
- Generation of the reports (Revenue and consumables) with getters, so the administrator can view them whenever they want.

The system does not include:

- The development of film/printing images.
- Payment system.
- The transfer of printed images/developed film from the photo studio to the customer.
- Replenishing the materials

Benefits:

- Quick distribution of orders
- Easy reporting/concise overview for administrator
- Automate the generation of the reports

Key features:

- Automatic price increase on orders in case of urgent orders
- Generation of different orders for the photographer and the client
- Generation of the reports

1.3 Use-Case Diagram

Provide a high-level UML use-case diagram showing main actors and their interactions with the system.

1.4 General Constraints

List technical and business constraints such as programming language, operating system, performance limitations, and standards.

1. Implementation language: C++
2. Platform: Linux and Windows desktop
3. Data storage: file system (text files)
4. Standards: UML notation for diagrams
5. Performance: An order must be created in ≤ 2 seconds
6. User interface: console only (CLI)

1.5 Assumptions and Dependencies

State assumptions (e.g., availability of internet, supported devices) and dependencies (e.g., external APIs, hardware).

1. - The photo studio has stable access to electricity, a local PC and to all other photo studio databases (user ID is incremented in all studios when creating a user, not just for one)
2. - The receptionist manually enters the client information into the system when creating the client, manually selects options and specifies completion time.
3. - The system depends on the local file system being accessible for storing transactions and reports.

1.6 Acronyms and Abbreviations

List all acronyms and abbreviations used in the document along with their explanations.

2

Terms Used

Description of terms

SW

Functional Requirements

2.1 Features / Functions to be Implemented

All functional requirements should be derived from User Stories or Use Cases.

This means that instead of listing abstract features, you first describe how users interact with the system and what goals they achieve.

User Stories – short, simple descriptions of a feature told from the perspective of the user (e.g., “As a registered user, I want to reset my password so that I can regain access to my account.”).

Use Cases – structured scenarios that describe interactions between actors and the system, including preconditions, steps, and outcomes.

From these stories/cases, you can then identify:

- *User interactions (e.g., authentication, profile management).*
- *Business processes (e.g., order processing, reporting).*
- *Integrations (e.g., with external APIs or third-party systems).*
- *System logic (e.g., validation, workflows, automation).*
- *Algorithms (if required, e.g., recommendation or prediction).*

Each function must be traceable back to a User Story or Use Case, ensuring that the system is built strictly according to user and business needs.

User stories:

- As the studio administrator, I want the photographer to have an efficient workflow with as minimal overhead as possible, so that the orders are completed more quickly.
- As the receptionist, I want to forward orders with accurate information so that I don't give out wrong orders.
- As the receptionist, I want to have orders generated within a matter of seconds, so that the customer and photographer don't have to wait long for them.
- As the receptionist, I want to automatically calculate daily income so that I don't report wrong numbers.
- As the photographer, I want to have clear deadlines so that I can complete orders in a sensible sequence.
- As the photographer, I want to have an automatic system to calculate daily resource usage so that I don't have to keep track of it and less mistakes happen.
- As the client I want to easily place my order so that getting my photo printed doesn't take me longer than it needs to.

From these stories, identifying functions:

User interactions:

- Reading user data
- Entering user data
- Entering order progress
- Placing order
- Reading an order

Business Processes:

- Order processing in a sensible sequence
- Reporting income
- Reporting completed orders
- Reporting used resources
- Payment

System logic:

- Automate prices calculation
- Automate generation of orders under five seconds
- Automate daily income sum calculation
- Automate daily resources used sum calculation
- Ensure the order has the correct deadline on it
- Client information and deadline are sent to the receptionist.

2.1 Acceptance Criteria

Define how each requirement will be validated: test cases, acceptance tests, or quality metrics.

In each photo studio a file for storing generated income and another file for storing spent resources must be created and appended to when the option is selected in the program.

The reports for total revenue generated and the total resources spent should match exactly the actual revenue and expenditure.

Orders must be correctly marked as express if the deadline is within 24 hours.

2.2 Implementation Requirements

Provide details of specific implementation requirements if applicable. For example, integration with existing systems, supported platforms, or algorithms.

3 SW Non-Functional Requirements

3.1 Resource Consumption

Specify performance and resource limits (CPU, memory, storage, response time).

Maximum memory usage: ≤ 100 MB

Maximum file size for daily logs: ≤ 5 MB

3.2 License Issues

State licensing requirements and constraints on third-party software or libraries.

Only standard C++ STL libraries are allowed.

No proprietary third-party libraries are permitted.

External libraries may only be used if they have permissive open-source licenses (MIT, Apache-2.0).

3.3 Coding Standard

Define coding style and standards that must be followed.

Each function and class has to include descriptive comments.

Tests have to cover all critical functions.

3.4 Modular Design

Specify architectural requirements such as modularity, extensibility, and maintainability.

The system should have separate modules for:

Placing the order

Payment

Recording customer details.

Passing order to client.

Passing orders to photographer

Price calculator.

Photographer reporting completion.

Photographer reporting consumed materials.

Receptionist reporting completed orders & revenue.

Overview of consumed materials & revenue by studio administrator.

3.5 Reliability

Define requirements for reliability, error handling, and fault tolerance.

3.6 Portability

List target platforms and environments where the system should operate.

The system must run on Linux.

Identical inputs must produce identical outputs on both platforms.

3.7 General Operational Guidelines

Provide guidelines for scalability, robustness, ease of use, and maintainability.

The system must be robust, easy to maintain, and simple to use.

All operations must be logged for accountability and auditing purposes.

4 SW Design Artifacts

4.1 CRC Cards (Class–Responsibility–Collaboration)

List the main classes with their responsibilities (action verbs) and collaborators (related classes); keep items concise and implementation-agnostic.

4.2 Conceptual UML Diagram (entities & relationships)

Draw a conceptual class diagram with key entities and their relationships; focus on nouns from User Stories/Use Cases, omit methods and low-level details.